Camera Estimation for Orbiting Pushbroom Imaging Systems

Rajiv Gupta and Richard I. Hartley

General Electric Corporate R&D
River Rd, Schenectady, NY 12309, USA

Abstract: Several space-borne cameras use pushbroom scanning to acquire imagery. Traditionally, modeling and analyzing pushbroom sensors has been computationally intensive due to the motion of the orbiting satellite with respect to the rotating earth, and the nonlinearity of the mathematical model involving orbital dynamics. The most time-consuming part of the computation involves mapping a 3-D point to its corresponding image point. A new technique for accomplishing this task that leads to fast convergence is described. In this technique, each iterative step assumes that the part of the orbital segment in which the satellite is operating is locally a straight line. It then moves the satellite so that the given 3-D point would come into the instantaneous view plane. As the satellite moves closer and closer to the destination point, this assumption becomes more and more valid, holding perfectly in the limit. In most cases we obtained the desired accuracy in one or two iterative steps. Thus, for most suitably initialized starting points, the model is linear. Besides computational efficiency, experimental results also confirm the accuracy of the model in mapping 3-D points to their corresponding 2-D points.

Keywords: Photogrammetry, Satellite camera models, Pushbroom sensors.

1 Pushbroom Sensors

The pushbroom principle is commonly used in satellite cameras for acquiring 2-D images of the Earth surface. In general terms, a pushbroom camera consists of an optical system projecting an image onto a linear array of sensors (Fig. 1). At any time only those points are imaged that lie in the plane defined by the optical center and the line containing the sensor array. This plane will be called the instantaneous view plane or simply *view plane* (see [1] for details).

This optical system is mounted on the satellite and as the satellite moves, the view plane sweeps out a region of space. The sensor array, and hence the view plane, is approximately perpendicular to the direction of motion. The magnitude of the charge accumulated by each detector cell during some fixed interval, called the *dwell time*, gives the value of the pixel at that location. Thus, at regular intervals of time 1-D images of the view plane are captured. The ensemble of these 1-D images constitutes a 2-D image. It should be noted that one of the image dimensions depends solely on the sensor motion.

SPOT satellite's HRV camera is a well-known example of a pushbroom system [2]. For HRV, the linear array of sensors consists of 6000 pixel array of electronic sensors covering an angle of 4.2 degrees. This sensor array captures a row pixels at 1.504 ms time intervals (i.e. dwell time = 1.504 ms). As the satellite orbits the earth, a continuous strip of imagery is produced. This strip is split into images, each consisting of 6000 rows. Hence a 6000×6000 pixel image is captured over a 9 seconds flight of the satellite. Such an image covers a square with side approximately 60 Km on the ground.

As a first level-of approximation, one is tempted to regard satellite imagery as aerial imagery from a very high platform. The following attributes, which are unique to satellite imagery, make this approximation rather crude [3].

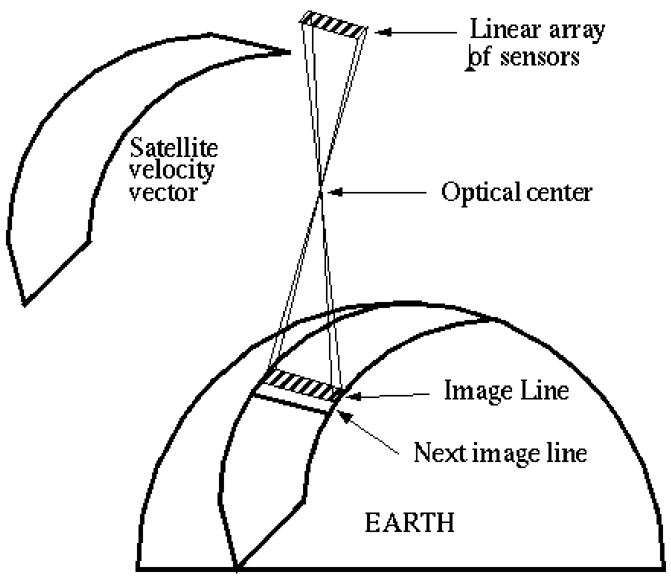

Figure 1: The pushbroom principle.

Many Perspective Centers. In pushbroom imagery, each line is imaged independently. This implies that there are numerous, highly correlated, perspective centers associated with each image.

Terrain Height to Altitude Ratio. Even for the satellites in low earth orbits, the terrain height to altitude ratio for satellite imagery is about 80 times smaller than that for aerial photography. Put another way, in satellite imagery, the terrain appears to be relatively flat. Thus the parallax arising due to difference in terrain relief and view angles in stereo pairs is not as pronounced. This implies that the parameters have to be known considerably more accurately if meaningful information is to be extracted from match point disparity.

Partially Perspective Image. In pushbroom imaging, image rays captured by the detector array are restricted to a plane perpendicular to the flight path. Thus the image is perspective only in the cross-flight direction; along the direction of flight, the projection is closer to being orthographic than perspective. A corollary of this fact is that the ray intersection method would not give accurate information about along-track location. In general, classical space resection techniques, by themselves, are not reliable enough for satellite imagery.

Field of View Angle. Generally the size of the detector array capturing the image is much smaller than its distance from the perspective center. This narrow field of view sometimes contributes to the instability of the solution.

Parameter Correlation. Scan line imaging suffers from high degree of parameter correlation. For example, any small rotation of the linear array around an axis perpendicular to the flight path (i.e., along the length of the array), can be compensated for by changing the position of the satellite along the orbital path. The maximum likelihood least-squares estimation model must be sufficiently constrained to take care of this problem.

Because of the above distinguishing features, it is well known that the standard photogrammetric bundle adjustment typical of aerial imagery does not work for satellite imagery [3, 1]. For accuracy, and in fact convergence, a pushbroom camera model must explicitly take into account the constraints imposed by: (1) the Kepler's Laws, (2) the rotation of the earth, and (3) the constraints imposed by the ephemeris data.

In order to ascertain quantitatively the mapping inaccuracy of the perspective approximation to a pushbroom image, we conducted the following experiment. We selected a grid of 3-D points with known latitude, longitude, and elevation. The 2-D positions of these grid points in a SPOT image was then derived using an exact model — the one that will be described in this paper — for the SPOT's pushbroom camera. To these ground control points, the best perspective camera (i.e., a pinhole camera) that minimizes the 3-D to 2-D mapping error was fitted. The 2-D mapping error (in pixels) for each grid point is shown in Fig. 2. As is clear from this figure, the error in modeling a pushbroom camera with a perspective camera can be as much as 40 pixels at the image boundary.

The effect of this modeling error in an application-specific context is illustrated in Fig. 3. This figure shows the digital elevation model (DEM) of a small tile of terrain derived from a stereo pair of SPOT images. Once again, both cameras were modeled as pin-hole cameras. An overall tilting of the derived terrain tile is quite apparent in Fig. 3. Since the ground truth was known for this example terrain tile, the corresponding error in terrain elevation is shown in Fig. 4. As can be seen, an attempt to model ortho-perspective imagery by a pinhole model can lead to discrepancy of more than 800 meters in the terrain model.

Even if one were to model the ortho-perspective nature of the imagery, classical space resectioning is unable to separate the correlation among the unknown parame-

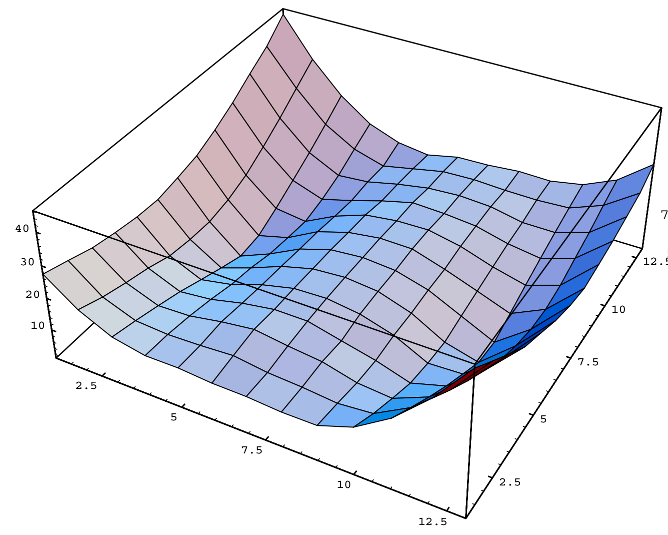

Figure 2: Mapping error in modeling a pushbroom camera as a pin-hole camera. The horizontal plane represents the image coordinates and the error, in pixels, is shown on the vertical scale.

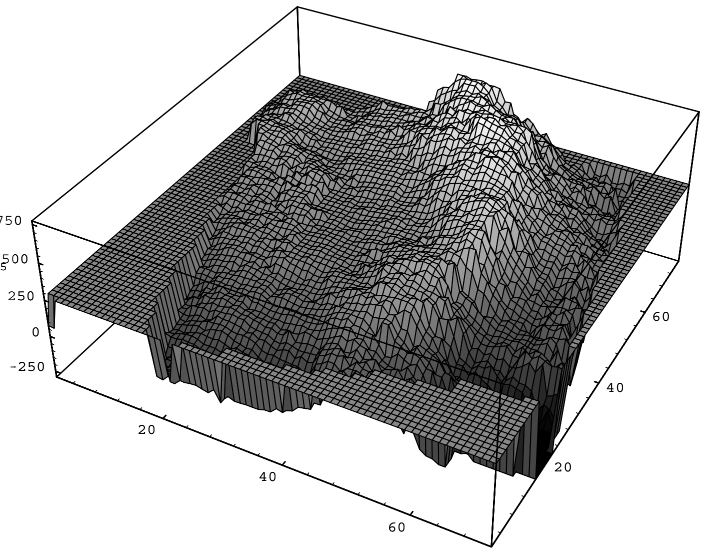

Figure 3: Terrain elevation estimated using pin-hole approximation for a SPOT stereo pair.

ters. Additional constraints are required in order to obtain convergence. For example, the constraint equations derived from known orbital relationships governed by Kapler's laws must be an integral part of the camera model. The task of modeling an orbiting pushbroom camera exactly must take into account the following factors.

- By Kepler's Laws, the satellite is moving in an elliptical orbit with the center of the earth at one of the foci of the ellipse. The speed is not constant, but varies according to the position of the satellite in its orbit. These position and velocity constraints must be imposed explicitly by the camera model.
- The earth is rotating with respect to the orbital plane of the satellite. The motion of the satellite with respect to the earth's surface should be incorporated in the camera model.
- The satellite is slowly rotating so that it is approximately fixed with respect to an orthogonal coordinate frame defined as follows: the z-axis emanates from the satellite and passes through the center of the earth; the x-axis lies in the plane defined by the satellite velocity vector and the z axis; the y-axis is perpendicular to the x and z axes. This coordinate frame will be called the *local orbital frame* (see Fig. 6 and Section 3). During one orbit, the local orbital frame undergoes a complete revolution about its y axis. This slow attitude change should be built into the camera model.
- The orientation of the satellite undergoes slight variations with respect to the local orbital frame. The camera model must account for this *attitude drift*.

Needless to say that for satellite cameras the task of finding the image coordinates of a point in space is relatively complex and computationally intensive because many of intermediate steps force the use of approximate or iterative schemes. For instance, there is no closed-form expression determining the angular position of an orbiting satellite given its time of flight from any given point in its orbit (e.g., time of flight from perigee). Because of this, the exact computation of the image produced by a pushbroom sensor has traditionally been a time consuming task.

This paper describes a general methodology for modeling a pushbroom camera that alleviates the problems mentioned above. A new technique for efficiently mapping a 3-D point to its corresponding 2-D image coordinate is described. Despite the non-linearity of the mathematical model, our scheme exhibits fast convergence.

In order to find the image coordinates of a given 3-D point, the satellite must be moved so that the 3-D point lies in its view plane (see Fig. 1). We present an iterative procedure for accomplishing this task. Each iterative step in this procedure assumes that the orbital segment between the current location of the satellite and its final

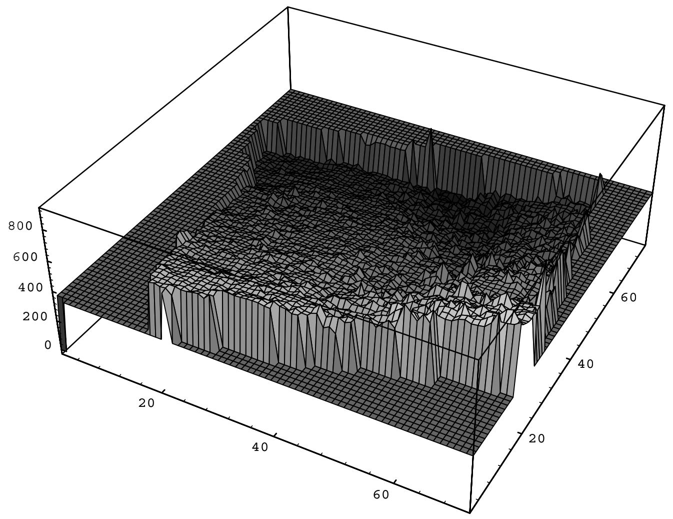

Figure 4: Error in the terrain generated using pin-hole approximation.

destination is a straight line. Under this assumption, the instantaneous velocity of the satellite is computed and it is moved so as to bring the given 3-D point into its instantaneous view plane. Clearly, the assumption about the linearity of the orbit is only an approximation; after this position update the satellite will be closer but not at its exact intended position. Repeated applications of this iterative step bring the satellite arbitrarily close to its destination position. In practice, the above operation is repeated until the angle between the instantaneous view plane and the ray to the 3-D ground point is less than some desired accuracy. We have found that this procedure has very fast convergence because as the satellite moves closer and closer to the destination point, the linearity assumption becomes more and more valid. In the limit the assumption holds perfectly and there is no approximation in the final answer that is obtained.

The model is computationally efficient: in most cases we obtained the desired accuracy in one iterative step. Thus, for most suitably initialized starting points, the model is linear. Experimental results also confirm the accuracy of the model in mapping 3-D points to their corresponding 2-D points.

The model described here has been implemented for the SPOT satellite's HRV cameras. Even though some of the terminology used refers specifically to SPOT, the model is applicable to all pushbroom cameras. To that extent, SPOT is just an example application.

2 Camera Model Estimation

The over all camera parameter estimation process can be divided into two main tasks, a modeling task and an optimization task.

2.1 Modeling Task

Camera modeling involves reconciling information from several different coordinate frames. For example, the ground points are generally given in a map coordinate system such as UTM or NAD-27; Ephemeris information is typically available in a geocentric coordinate system; computations involving look angles are most convenient in the local orbital frame. For accurate camera model estimation, it is essential that these frames be precisely registered with each other. How this is done in our system will be described in this and the next sections.

Before we can estimate the parameters of a camera, we have to implement a software model of the camera. This software model transforms a point in the world coordinate system (given, for example, as [lat, lon, elevation]) into a pixel location (u, v) of the same point, in accordance with parameters and mechanisms of the camera. A camera modeling routine essentially mimics the operation of the camera as it transforms a ground point into a pixel in the image.

Since the main camera modeling task is to map a given point from the world coordinate system to the image coordinate system we define these two coordinate systems below.

Image Coordinates. Each pixel in the image is identified by its row and column numbers. The rows of an image represent the satellite motion: successive rows are imaged one dwell time apart. Thus by selecting a suitable origin, a time stamp can be associated with each row in the image. The columns represent the field-of-view in the cross flight direction.

World Coordinates. As mentioned earlier, a ground point (i.e., its latitude, longitude, and elevation), the auxiliary information provided by the on-board systems (e.g., the ephemeris information), and the calibration information (e.g., the look angles) may all be with respect to three different world coordinate systems. To simplify processing, it is useful to transform all available information to a geocentric, geo-fixed frame. In our implementation, we have chosen to convert all parameters to the 1980 Geodetic Reference System (GRS-80) since the ephemeris information supplied on a SPOT tape are provided in this coordinate system. A different coordinate frame may be more suitable for another camera type.

GRS-80 is an earth centered, earth fixed, rotating coordinate system. In this coordinate system, the X-axis passes through the Greenwich meridian, the Z-axis coincides with the axis of the earth's revolution, and the Y-axis is perpendicular to both the X and Z axes so as to form a right-handed system. GRS-80 models earth as an ellipsoid with semi-major and minor axes given by $R_a = 6378.137$ km and $R_b = 6356.7523$ km, respectively.

Given the geocentric latitude (ψ_s) and longitude (λ_s) of the satellite, and its orbital radius (R_s), one can find its GRS coordinates using the following formula.

$$\begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} = \begin{bmatrix} R_s \cos\psi_s \cos\lambda_s \\ R_s \cos\psi_s \sin\lambda_s \\ R_s \sin\psi_s \end{bmatrix} \tag{1}$$

One can also compute geodetic latitude (ϕ_e) and longitude (η_e) from the geocentric latitude (ψ) and longitude (λ) as follows:

$$\eta_e = \lambda_e \tag{2}$$

$$\phi_e = \tan^{-1}(\frac{R_a^2}{R_b^2}\tan\psi_e) \tag{3}$$

From the *geodetic* latitude (ϕ_e), longitude (η_e), and elevation (h) of a point on the earth's ellipsoid, the GRS-80 coordinates can be computed using the following equations.

$$\begin{bmatrix} X_e \\ Y_e \\ Z_e \end{bmatrix} = \begin{bmatrix} (N\cos\psi_e + h\cos\phi_e)\cos\eta_e \\ (N\cos\psi_e + h\cos\phi_e)\sin\eta_e \\ (N\sin\psi_e + h\sin\phi_e) \end{bmatrix} \tag{4}$$

where N is given by

$$N = \frac{R_a R_b}{R_b^2\cos^2\psi_e + R_a^2\sin^2\psi_e}. \tag{5}$$

The above equations are sufficient to transform all input data to GRS-80.

2.2 Optimization Task

A ground control point is defined as a 3-D point whose image coordinates are known. One can assemble a set of ground control points by picking prominant features on a map — e.g. road intersections, buildings, or other geographically distinguished points on the ground — and noting their 2-D location in the image for which we want to compute the camera model. The optimization task takes a set of ground control points as input and results in an estimate of the camera parameters.

If a routine to transform any GRS-80 point (x, y, z) into its image coordinates (u, v) is available, one can formulate the camera parameter estimation problem as a global optimization problem. In this formulation, the parameters of the camera are the unknown variables while the *ground control points* and the ephemeris information collected by the on-board systems provide inputs and constraints. The over all task is to compute a set of camera parameters that minimize the least-squared-error between the given and computed pixel values for each ground control point while abiding by all the orbital constraints.

The optimization method used in our implementation is based partly on the well known Levenberg-Marquardt (LM) parameter estimation algorithm. Our extensions to the basic LM algorithm include methods for handling sparsity in the Jocobian matrix and its customization for the camera parameter estimation problem. This implementation is briefly described in Section 6 and detailed in [4].

The following section discusses the various parameters associated with a satellite mounted pushbroom camera and classifies them.

3 Model Parameters for a Pushbroom Camera

All the parameters needed for modeling the camera have a predetermined nominal value which is known prior to the launch. For some parameters, since they are continuously monitored by on-board systems, a more accurate value is provided to the user as ephemeris and other auxiliary information. Nevertheless, for the sake of greater accuracy it has proven necessary to refine the ephemeris data and estimate all the parameters simultaneously by solving the overall mapping problem using ground-control points and orbital constraints.

Camera model for an orbiting pushbroom camera uses a large number of parameters. It is useful to classify the camera parameters into three classes: *known parameters*, *independent parameters*, and *dependent parameters*. At the implementation level, the parameter information in the camera modeling software is distributed among three structures called *known_params*, *dependent_params* and *independent_params*. Most routines are simply passed a

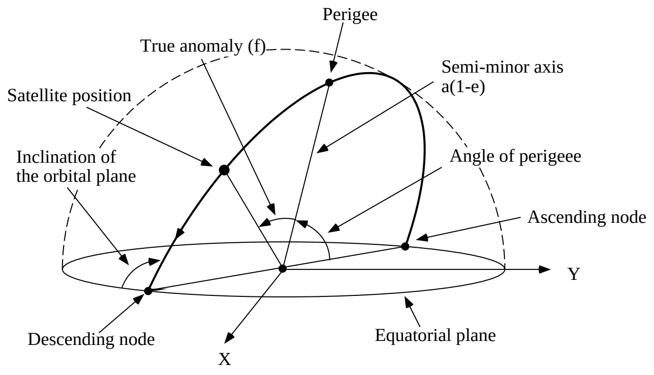

Figure 5: The six orbital parameters.

pointer to these structures. The contents of these structures are described below.

Independent Parameters. The exact position of a satellite in its orbit is fully described by the following six parameters which are illustrated in Figure 5 (also see [5]):

1. semi-major axis of the orbital ellipse (a),
2. orbital eccentricity (e),
3. inclination of orbital plane with respect to the equatorial plane (i),
4. geocentric angle between the perigee and the ascending node (ω),
5. longitude of the ascending (or descending) node (λ_{AN_k} or λ_{DN_k})), and
6. true anomaly (f).

The first two parameters determine the elliptical shape of the orbit. The third and fourth parameters fix this ellipse in space with respect to the equatorial plane. The fifth parameter, λ_{AN_k} (or, equivalently, λ_{DN_k}), registers the k-th orbital track with the rotating earth. In this list, f is the only time dependent parameter; all others can be assumed to be fixed for any given track of the satellite.

Because of the rotation of earth, the equator crossing of the satellite drifts westward with each revolution. Let λ_{DN_1} the longitude of the first equator crossing when the satellite is moving from north to south (also known as the longitude of the first descending node). The nominal values for the longitude of the descending node, ascending node and the top of the orbit for any given revolution

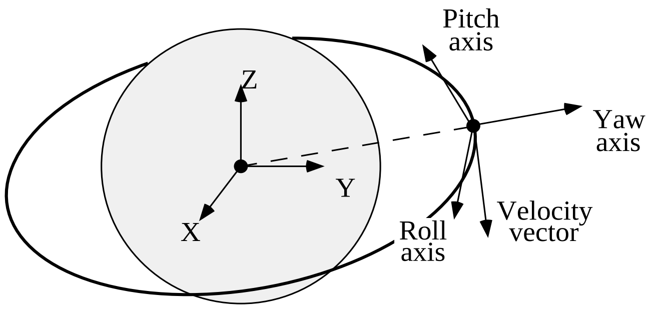

Figure 6: The local orbital frame.

number can be calculated using the following relationships.

$$\begin{aligned} \lambda_{DN_k} &= \lambda_{DN_1} - (k-1) \times \omega_e \times T(360^\circ) \quad (6) \\ \lambda_{AN_k} &= \lambda_{DN} + 180^\circ + \omega_e \times T(180^\circ) \\ \lambda_{T_k} &= \lambda_{AN_k} - 90^\circ - \omega_e \times T(90^\circ) \end{aligned}$$

Here, $T(x)$ is the time the satellite takes to negotiate an angle x from the ascending node, and w_e is the angular speed, in degrees/second, of Earth's rotation.

The above orbital parameters specify the position of the camera platform. In order to specify the orientation or the pose of the camera the following reference frames are needed.

A **Local Orbital Frame** is defined at every point in the orbit as follows (see Fig. 6). The origin of the frame is the satellite's center of mass; the *yaw axis* is the geocentric vector pointing radially away from the Earth center; the *roll axis* is in the orbital plane perpendicular to the yaw axis, along the velocity vector; and *pitch axis* is perpendicular to both yaw and roll axes.

The **Satellite Attitude Reference Frame** is fixed with the satellite. Nominally it is aligned with the local orbital reference frame as follows: the X axis is along the pitch axis, the Y axis is aligned with the roll axis and the Z axis is aligned with the yaw axis. The angles between the attitude frame and local orbital plane are used to orient the satellite.

The complete orientation of the satellite is computed in two parts: (1) the attitude or the *look direction* of each pixel in the detector array within the satellite attitude reference frame, and (2) the orientation of the attitude reference frame with respect to the local orbital reference frame.

First we specify the *look direction* of each detector element. It is customary to specify the look direction by two angles: Ψ_x and Ψ_y (Fig 7). Ψ_x represents the rotation that causes the satellite to look forward or backward along the direction of flight; Ψ_y is the rotation perpendicular to it. More precisely, the first angle Ψ_x is the angle made by the orthogonal projection of the look direction in the Y-Z plane with the negative Z axis of the satellite attitude reference frame. If the camera is pointed towards the nadir, this angle is zero; a non-zero Ψ_x makes the satellite look forward or backward along the ground track. Similarly, Ψ_y is the angle that the orthogonal projection of the look direction vector, projected in the X-Z plane, makes with the negative Z axis. In nadir viewing, Ψ_y is zero for the central pixel; it gradually increases for detectors looking eastward, and decreases for detectors looking westward (see [2]).

Given Ψ_x and Ψ_y, the unit vector along the look direction in the attitude reference frame is given by $U = [\tan\Psi_y, \tan\Psi_x, 1]/\sqrt{1+\tan^2\Psi_x+\tan^2\Psi_y}$. The look direction of the pth pixel in the attitude reference frame can be computed from that of the first and the Nth pixel by interpolation using $U_p = (1-\frac{p-1}{N-1})U_1 + (\frac{p-1}{N-1})U_N$, where U_1 and U_N are the look directions vectors for the first and the N-th pixels.

The orientation of the satellite attitude reference frame can be specified by three rotation angles, $RotX_i$, $RotY_i$, and $RotZ_i$, of the attitude frame with respect to the local orbital frame, for each row i in the image. Nominally, $RotX_i$, $RotY_i$, and $RotZ_i$ are zero at every point in the orbit. These parameters are continuously monitored by the attitude control system and their rate of change (instead of the actual value) is reported as a part of the auxiliary information gathered during image acquisition.

We assume that $\frac{d(RotX_i)}{dt}$, $\frac{d(RotY_i)}{dt}$, and $\frac{d(RotZ_i)}{dt}$ are available for each row, either directly, or through interpolation. Under this assumption, the drift of the attitude frame with respect to the local orbital plane can be computed by integration if that for the first row of imagery is known. This gives rise to three new independent variables $RotX_1$, $RotY_1$, and $RotZ_1$, the rotation angles for the first row.

Besides the above parameters, other independent camera parameters include (1) time from the perigee to the scene center, (2) the dwell time for the detectors (i.e. the time between image lines), and (3) the image coordinates of the center pixel.

Dependent Parameters. These parameters can either be computed from the independent parameters or are measured directly by the on-board systems. The list includes such measurable and given parameters as the ephemeris information (positions and velocities at different points in the orbit), ground control points (i.e., associations between [lat, lon, elevation] and (u, v) in the image), and rates of change of $RotX_i$, $RotY_i$, and $RotZ_i$. Dependent parameters are used to impose constraints on the solution.

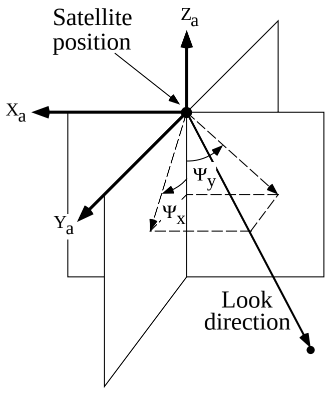

Figure 7: Look angles Ψ_x and Ψ_y.

4 Tracking the Satellite

A satellite provides a stable and, more importantly, a predictable platform. Thus one can employ constraints dictated by the Kepler's laws to achieve convergence. In particular, the following laws can be used to position a satellite at any desired location in the orbit, given the value of its independent parameters.

1. The orbits are elliptical.
2. The vector from earth's center to the satellite sweeps equal area in equal time.
3. The time period P is given by $P^2 = \frac{4\pi a^3}{GM_e}$, where a is the semi-major axis of the orbital ellipse and GM_e is the gravitational constant times mass the mass of the earth.

This section details the procedures for computing the angular position of a satellite given its travel time from perigee and vice versa. The definition of the various elliptical parameters such as true and eccentric anomalies, and relationships among them, essential for the derivations that follow, are given in Figure 8.

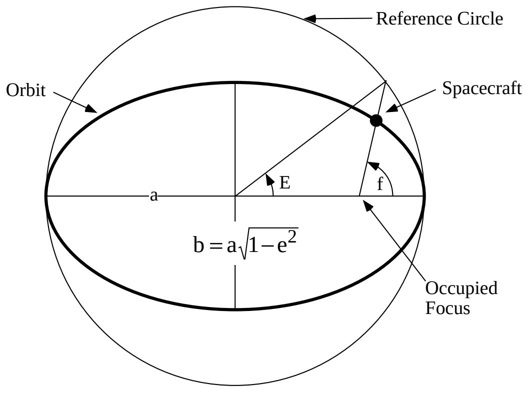

Figure 8: Parameters of an ellipse.

Computation of Elapsed Time from True Anomaly.

The true anomaly f can be converted into the eccentric anomaly E using,

$$\cos E = \frac{(\cos f + e)}{(1 + e\cos f)} \tag{7}$$

where e is the eccentricity of the orbit. A direct relationship exists between the mean anomaly M and the eccentric anomaly E.

$$M = E - e\sin E \tag{8}$$

From Kepler's second law, which relates mean anomaly and mean angular velocity ($\bar{\omega}$), one can compute the elapsed time as $t = M/\bar{\omega}$. In this equation, $\bar{\omega}$ can be computed using Kepler's third law: the mean angular velocity ($\bar{\omega}$) is given by:

$$\bar{\omega} = \sqrt{\frac{GM_e}{a^3}} \tag{9}$$

where a is the semi-major axis of the orbit, G is the universal gravitational constant, and M_e is the mass of the earth.

Computation of True Anomaly from Elapsed Time

One is tempted to back-trace the computation described above to compute the true anomaly from the elapsed time. However, Eq. 8 that relates M and E is not explicit in M. To overcome this one must either linearize it to express E in terms of M, or compute E iteratively.

Once again, Mean Angular Velocity can be computed from the Kepler's 3rd law, $\bar{\omega} = \sqrt{(}GM_e/a^3)$. The mean anomaly, then, is given by $M = t \times \bar{\omega}$, where t is the elapsed time. One can first compute, from mean anomaly M, a rough value for the eccentric anomaly E using a series expansion of $M = E - e\sin E$:

$$\begin{aligned} E = M \quad & + \quad (e - e^3/8 + e^5/192 - e^7/9216)\sin M \\ & + \quad (e^2/2 - e^4/6 + e^6/98)\sin(2M) \\ & + \quad (3e^3/8 - 27e^5/128 + 243e^7/5120)\sin(3M) \\ & + \quad (e^4/3 - 4e^6/15)\sin(4M) \\ & + \quad (125e^5/384 - 3125e^7/9216)\sin(5M) \\ & + \quad (27e^6/80)\sin(6M) \end{aligned} \tag{10}$$

With this coarse value as the starting point, we can solve for fixed point iteratively as follows.

```
old_E = 0.0;
while(fabs(E - old_E) > EPSILON) {
   old_E = E;
   E = M + e * sin(old_E); }
```

The value of E computer using Eq. 10 is generally quite close to its final value. In practice, the above iteration rarely takes more than one or two iterative steps. Finally, to compute the true anomaly f from eccentric anomaly E, we use the identity

$$\tan f = \frac{\sqrt{1-e^2}\sin E}{\cos E - e}. \tag{11}$$

5 The Mapping Algorithm

Fig. 9 depicts the mapping of a 3-D ground point to its corresponding image point. The satellite's initial position S in the orbit at time $t = 0$ is marked A. At this time instant, everything in the intersection of the view plane ABC and the ground swath is observed by the satellite. The satellite will observe the point (x, y, z) from the point D in the orbit: at this point, the new view plane (DEF) passes through the ground point. In order to estimate the u image coordinate, i.e. the image coordinate of (x, y, z) that depends on time, the satellite must be moved to point D from a known starting point such as A. We accomplish this by making the dihedral angle between the view plane and the ray SX equal to zero, where S denotes the instantaneous position of the satellite in the orbit.

The pseudo-code for transforming a 3-D point `pt3d` to its 2-D image coordinates `u` and `v` — in essence moving the satellite from A to D — is given below. All the camera parameters are stored in four structures `kp`, `ip`, `dp`, and

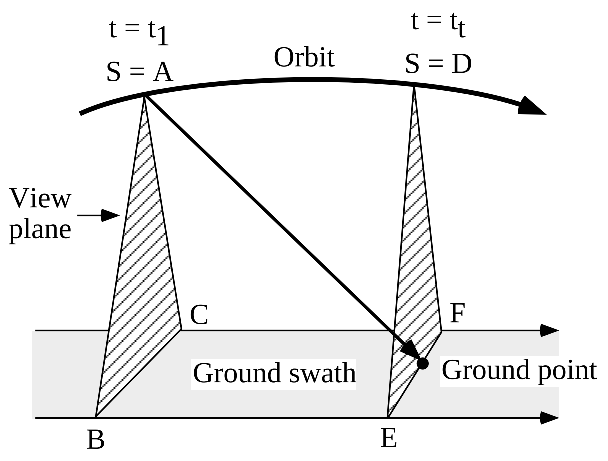

Figure 9: Bringing the ground point in the view plane.

ap which contain the known, independent, dependent, and auxiliary parameters described earlier. The syntax s->m is used to denote a variable m stored in the structure s.

```
void transform(pt3d, u, v, kp, ip, dp, ap)
{
  /* STEP 1. */
  /* Find the true anomaly of the
     scene center */
  anomaly = time2anomaly(kp, ip,
            ip->center_travel_time);

  /* Move the satellite to the
     scene center */
  anomaly2parameters(kp, ip, dp, anomaly);

  /* STEP 2. */
  /* Enter coarse pointing mode */
  disable_drift_calculations();

  /* Compute the viewing angles */
  compute_angles (pt3d, kp, ip, dp, ap);

  /* Move the satellite such that the
   * look vector is in the view plane */
  closein_angle_discrepancy(0.0, pt3d, kp,
                            ip, dp, ap);

  /* STEP 3. */
  /* Enter fine pointing mode */
  enable_drift_calculations();

  /* Recompute the viewing angles */
  compute_angles (pt3d, kp, ip, dp, ap);

  /* Close-in the angular discrepancy
     with drift computation enabled */
  closein_angle_discrepancy(0.0, pt3d, kp,
                            ip, dp, ap);

  /* STEP 4. */
  /* Deduce u and v from travel_time and
     viewing_angle_ratio */
  u = ip->Ppu +
      (dp->travel_time -
       ip->center_travel_time)/ip->dwell_time;
  v = 1 + 2*(ip->Ppv - 1)*
          dp->viewing_angle_ratio;
}
```

The algorithm executes the following steps.

1. Initialization. The 3-D to 2-D mapping algorithm always starts at the scene center (i.e., the point A in the above description is the point in the orbit where the central row of the image was acquired). The true anomaly of A is computed using the time from perigee to the scene center and the satellite is moved there. Recall that the time from perigee to the scene center is an independent parameter; Computation of anomaly, given the travel time, has already been detailed in Section 4. The routine anomaly2parameters computes all the dependent parameters of the satellite at position A in the orbit. In particular, the following values are computed and returned as dependent parameters:

1. satellite's orbital position coordinates and its velocity vector in GRS,
2. distance of the satellite and its nadir point from the geo-center,
3. latitude and longitude of the nadir point,
4. satellite heading (i.e. angle between its motion vector and the longitude),

The satellite is moved from point A to D in two main steps called the *coarse* and *fine pointing modes*.

2. Move the View Plane: Coarse Pointing Mode. In this mode it is assumed that the satellite is a perfectly stable platform and any drifts in its attitude are ignored by disabling the drift calculations using disable_drift_calculations(). In other words, it is assumed that the local orbital frame and the satellite attitude frame are assumed to be perfectly aligned with each other at every point in the orbit.

Consider the angle between the view plane and the ray from the satellite to the ground point. At point D in the orbit, the point at which the ground point would be imaged, this angle would be zero. So the algorithm attempts to move the satellite to a place in the orbit such that the ground point is in the viewing plane. This task is accomplished by the routine `closein_angle_discrepancy` as follows.

Assume that the satellite is flying in a straight line as shown in Fig. 10. Let the instantaneous position of the satellite be $t = t_1$ as shown. At this time instant, one can compute the angle between the ray and the view plane in a straight-forward manner. Instead of working with the angle discrepancy between the look direction and the view plane, we work with its complement, viz., the angle between the look direction and the direction of the motion of the satellite. We want to move the satellite to its target position at $t = t_t$ where the angle is Θ_t. In order to accomplish this, we them move the satellite a small time step Δt to a new position t_2. At this new time instant, we recompute the position, the velocity vector, and the angle Θ_2. Using sine law,

$$\begin{aligned} \frac{\Delta t}{\sin(\Theta_2 - \Theta_1)} &= \frac{\chi}{\sin \Theta_1}, \quad and \\ \frac{\delta t}{\sin(\Theta_t - \Theta_2)} &= \frac{\chi}{\sin(180 - \Theta_t)}. \end{aligned} \tag{12}$$

Eliminating the unknown χ, we get

$$\delta t = \frac{\sin(\Theta_t - \Theta_2)}{\sin \Theta_t} \frac{\sin \Theta_1}{\sin(\Theta_2 - \Theta_1)} \Delta t. \tag{13}$$

It can be shown that no matter where t_2 is with respect to t_1 and t_t, the above equation holds. Thus we can bring the satellite to bear any desired look angle Θ_t. In practice, when the satellite is actually moved by $\Delta t + \delta t$, the angular discrepancy becomes smaller but is not exactly zero. This is because of the assumption concerning the straight-line motion of the satellite. The above step is repeated till the angular discrepancy is as close to zero as is numerically meaningful.

3. Move the View Plane: Fine Pointing Mode. In this mode, the attitude drifts measured by on-board system are taken into account using `enable_drift_calculations()`. Once again the viewing angle is computed iteratively and the satellite is moved to a place in the orbit such that `pt3d` is in the viewing plane. The only difference between the coarse and fine pointing modes is that at each iterative step the angles computed are modified by the drift in satellites attitude. To do this, the drift in yaw, roll and pitch angles of the attitude frame with respect to the local orbital frame are determined as follows.

Typically, the rate of angular change in the satellite's orientation is measured by the on-board systems and

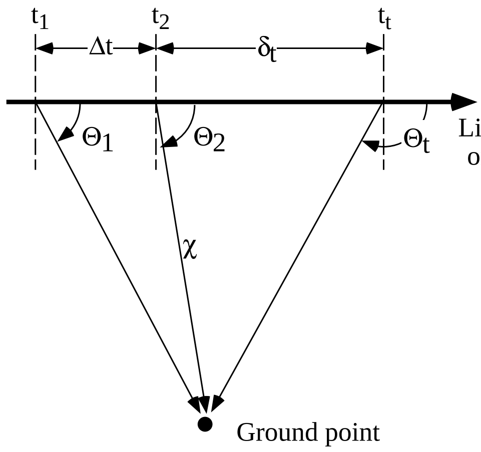

Figure 10: Angular discrepancy and satellite motion.

given as a part of the ephemeris information. From the few select points at which these rates of change of yaw, roll and pitch angles are given, the rate of angular drift for each image row is determined by interpolation. From this data, the actual drift along yaw, roll and pitch axes is then computed via integration.

4. Computation of u and v. Once the satellite has been brought to place where the 3-D ground point lies in the instantaneous view plane, the travel time contains the information about the u coordinate. Let Ppu and Ppv denote image center.

$$u = Ppu + (travel\_time \; - \; center\_travel\_time)/dwell\_time \tag{14}$$

Within the instantaneous view plane, the fraction of field of view angle, in the cross-flight direction, that is being cut by the look direction needed to see `pt3d` contains the information about the v coordinate. Let $viewing\_angle\_ratio$ denote this fraction. The v coordinate is given by

$$v = 1 + 2(Ppv - 1) viewing\_angle\_ratio \tag{15}$$

The above iterative procedure always starts at the scene center. Since most of the times the computed image coordinate lies close to the one computed previously, it would appear that one can make the above scheme more efficient by starting the procedure at the last position of the satellite instead of the scene center. However, in practice, this reduces the number of iterations by at most one and is not a significant time saver.

6 Optimization Task

The algorithm described in the previous section gives us a routine $transform(kp, ip, dp, ap)$ that, for given values of the independent camera parameters ip, maps a 3-D point (x, y, z) to its corresponding image point (u, v). In practice, we are given ground control points — i.e., 3-D points on the ground whose image coordinates are know a priori — and we want to estimate the values in ip. If a sufficient number of ground control points are given, the values of the independent parameters in ip can be computed using the *collinearity condition*. This condition states that a ray that originates from the instantaneous projection center of the camera and passes through the point (u, v) in the image plane, must also pass through the point (x, y, z). Thus, for an initial value of ip, we can affect the mapping

$$transform(kp, ip, dp, ap) : (x_i, y_i, z_i) \rightarrow (\hat{u}_i, \hat{v}_i)$$

and iteratively modify ip until the RMS re-projection error

$$\sum_{i=1}^{n} \sqrt{(u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2} \quad (16)$$

is minimum.

If several images of the same scene are available, all cameras can be solved simultaneously by enforcing additional *coplanarity constraints* derived from image to image match points. Let (u_i, v_i) and (u'_i, v'_i) be a pair of corresponding points in a stereo pair and let the associated cameras be denoted by (kp, ip, dp, ap) and (kp', ip', dp', ap'). The rays emanating from (u_i, v_i) and (u'_i, v'_i) must meet in space at some unknown 3-D point. In other words, there exists a point (x_i, y_i, z_i) such that

$$\begin{aligned} transform(kp, ip, dp, ap) &: (x_i, y_i, z_i) \rightarrow (u_i, v_i) \quad (17) \\ transform(kp', ip', dp', ap') &: (x_i, y_i, z_i) \rightarrow (u'_i, v'_i) \quad (18) \end{aligned}$$

From the initial values of the camera parameters, we can compute an initial value of (x_i, y_i, z_i). Regarding (x_i, y_i, z_i) also as a parameter to be determined, we can include the match points (u_i, v_i) and (u'_i, v'_i) in the objective function given in Eq. (16).

The Levenberg-Marquardt (LM) algorithm, a well known algorithm for parameter estimation ([6]), was used to find the optimal values of camera model parameters that minimize (16). In general terms, given a hypothesized functional relation $\mathbf{y} = f(\mathbf{x})$ where $\mathbf{x}$ and $\mathbf{y}$ are vectors in some Euclidean spaces R^m and R^n, and a measured value $\hat{\mathbf{y}}$ for $\mathbf{y}$, the LM algorithm computes the vector $\hat{\mathbf{x}}$ that most nearly satisfies this functional relation. Our implementation of the LM algorithm is described in detail in see [4] and will not be repeated here.

Based on the implementation of the LM algorithm in [6], we have coded a general minimization routine. To use this algorithm in the simplest form it is necessary only to provide a routine to compute the function f being minimized, a goal vector $\hat{\mathbf{y}}$ of observed or desired values of the function and an initial estimate $\mathbf{x}_0$. In this case, f corresponds to the transform routine presented earlier, the goal vector $\hat{\mathbf{y}}$ is $(\hat{u}_i, \hat{v}_i)$ so that (16) is minimized, and initial estimate for $\mathbf{x}_0$ comes from the nominal values of the SPOT parameters.

If desired, it is possible to provide a function to compute the Jacobian matrix J. If a null function is specified, then the differentiation is done numerically. Finding derivatives with respect to the model variables in the orbital equations is a non-trivial exercise. For this reason, we did not specify a Jacobian matrix and used numerical differentiation, which is carried out as follows. Each independent variable x_i is incremented in turn to $x_i + \delta$, the resulting function value is computed using the routine provided for computing f and the derivative is computed as a ratio. The value δ is set to the maximum of $|10^{-4} * x_i|$ and 10^{-6}. This choice seemingly gives a good approximation to the derivative. In practice, we have seen almost no disadvantage in using numerical differentiation.

As an alternative to all the dependent variables being equally weighted, it is possible to provide a weight matrix specifying the weights of the dependent variables $\mathbf{y}$. This weight matrix may be diagonal specifying independent weights for each of the y_i, or else it may be symmetric, equal to the inverse of the covariance matrix of the variables y_i.

At the start of parameter estimation, most parameters are initialized to their nominal or auxiliary value and an appropriate standard deviation is specified for each. The values of the ephemeris information and the control points are given to put the constraint. The following section describes the results of a typical experimental run.

7 Experimental Results

A pair of SPOT stereo images of the Los Angeles area were used to calibrate the corresponding cameras. A set of 25 ground control points and 100 image to image match points were used for calibration. The algorithm

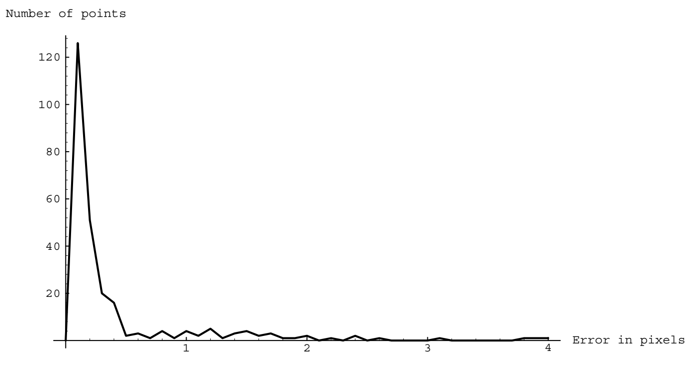

Figure 11: Error in re-projected points.

took about 1 minute on a SPARC 10 to solve for both cameras.

Fig. 11 shows these points in the first image. The 3-D points projected using the computed camera are also shown in Fig. 11 along with the given points. Because the camera can map a 3-D point to its corresponding 2-D point with sub-pixel accuracy, all point pairs in Fig. 11 appear as a single point.

Fig. 11 shows the error distribution of the reprojected points. As can be seen, about 90% of the points have a reprojection error of less than 1.0 pixel and over 95% are with in 2 pixel error. Points with larger than two pixel errors were manually confirmed to be outliers arising from errors in the automatic matching procedures (i.e., these point pairs were mistakenly identified as match points). The overall RMS error with which a ground point can be mapped to its corresponding image point it 0.73 pixels.

Parameter	Actual	Nominal
Semi-major axis a	7182980	7203322 m
Eccentricity e	0.00103919	0.001327417
Inclination i	98.77000	98.73727 deg
Perigee angle w	90.00000	71.39493 deg
Longitude of DN	-131.31380	-131.2472 deg
Look angle Ψ_{x1}	0.54666602	0.8728483 deg
Look angle Ψ_{xn}	0.56695408	0.8895469 deg
Look angle Ψ_{y1}	0.22989154	0.2299154 deg
Look angle Ψ_{yn}	27.1112440	27.11084 deg
Time perigee to ctr	1238.47153	1237.909 sec
Dwell time	0.00150400	0.001503339 sec

Table 1: Estimated vs. nominal parameter values.

Table 1 shows the estimated independent camera parameters for the first camera along with their nominal values. The parameter values for the second camera are similar and skipped for brevity. As can be seen, a considerable variation exists between the actual and the nominal values. If the nominal values for independent parameters were to be used for 3-D to 2-D mapping, we would get an RMS error of 48.92 pixels. On the ground, this is equivalent to having a position discrepancy of about 489 meters.

References

[1] Rajiv Gupta. Simple camera models for orbiting pushbroom sensors. In *The proceedings of 1995 Mobile Mapping Symp*, The Ohio State Universiy, May 1995.

[2] SPOT Image Corporation, 1897, Preston White Dr., Reston, VA 22091-4368. *SPOT User's Handbook*, 1990.

[3] Ashley P. Tam. *Terrain elevation extraction from digital SPOT satellite imagery.* PhD thesis, Masters Thesis, Dept. of Surveying Engineering, Calgary, Alberta, July 1990.

[4] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. In *Proc. of the Second Europe-US Workshop on Invariance, Ponta Delgada, Azores*, pages 187–202, October 1993.

[5] C. C. Slama, editor. *Manual of Photogrammetry.* American Society of Photogrammetry, Falls Church, VA, fourth edition, 1980.

[6] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1988.

[7] R. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 761–764, 1992.

[8] Richard I. Hartley and Rajiv Gupta. Linear pushbroom cameras. In *Computer Vision - ECCV '94, Volume I, LNCS-Series Vol. 800, Springer-Verlag*, pages 555–566, May 1994.

[9] R. Gupta and R. Hartley. STEREOSYS 3.0 user's guide. Report, GE Corporate R&D, June 1992.

[10] R. Gupta, R. Hartley, and J.L. Mundy. Experimental verification of the accuracy of projective, perspective, and linear pushbroom cameras. Internal report, iu group, GE Corporate R&D, June 1993.

[11] R. Gupta and R. Hartley. Camera estimation for orbiting pushbrooms. In *The Proc. of Second Asian Conf. on Computer Vision*, volume 3, Singapore, December 1995.

[12] M. J. Hannah. Bootstrap stereo. In *Proc. Image Understanding Workshop, College Park, MD*, pages 210–208, April 1980.

[13] M. J. Hannah. A description of SRI's baseline stereo system. Technical Report Tech. Note 365, SRI International Artificial Intelligence Center, Oct. 1985.

[14] R.N. Colwell, editor. *Manual of Remote Sensing.* American Society of Photogrammetry, Falls Church, VA, second edition, 1983.

[15] Rajiv Gupta and Richard I. Hartley. Linear pushbroom cameras. September 1997.